

# Automation of a Navier-Stokes S&C Database Generation for the Harrier in Ground Effect

Scott M. Murman\*  
ELORET  
Moffett Field, CA 94035  
smurman@nas.nasa.gov

Neal M. Chaderjian† and Shishir A. Pandya‡  
NASA Ames Research Center  
Moffett Field, CA 94035

## Abstract

A method of automating the generation of a time-dependent, Navier-Stokes, static stability and control (S&C) database for the Harrier aircraft in ground effect is outlined. Reusable, lightweight components are described which allow different facets of the CFD simulation process to utilize a consistent interface to a remote database. These components also allow changes and customizations to easily be facilitated into the solution process to enhance performance, without relying upon third-party support. An analysis of the multi-level parallel solver OVERFLOW-MLP is presented, and the results indicate that it is feasible to utilize large numbers of processors ( $\approx 50$ ) even with a grid system with a relatively small number of cells ( $\approx 10^6$ ). Using the tools described in this paper, improvements in both wallclock and computational time to generate a database of time-dependent solutions have been observed.

## 1 Introduction

The Harrier is a V/STOL aircraft that can take-off and land vertically, or utilize very short runways, by directing its four exhaust nozzles towards the ground. Transition to forward flight is achieved by rotating these nozzles into a horizontal position. Powered-lift vehicles such as the Harrier have certain advantages over conventional aircraft. Their V/STOL capabilities allow for safer carrier operations, smaller carrier size, and allow for quick reaction time for troop support. They also are not dependent on vulnerable land-based runways. The AV-8A was the



Figure 1: Harrier AV-8B in hover (Boeing).

first service-version of the Harrier, and the AV-8B was a later redesign for improved payload capacity and range (cf. Siuru [1]). The current work utilizes a version of the AV-8B design. The success and unique capabilities of the Harrier has prompted the design of a powered-lift version of the Joint Strike Fighter (JSF).

The flowfield for the Harrier near the ground during very low-speed or hover flight operations is very complex and time-dependent (cf. Fig.1). Warm air from the fan is exhausted from the front nozzles, while a hot air/fuel mixture from the engine is exhausted from the rear nozzles. These jets strike the ground and move out radially forming a ground jet-flow. The ambient freestream, due to low-speed forward flight or headwind during hover, opposes the jet-flow. This interaction can cause the flow to separate and form a ground vortex which can be unsteady, changing size and position at low frequency. The multiple jets also interact with each other near the ground and form an upwash, or jet fountain, which strikes the underside of the fuselage. If the aircraft is sufficiently close to the ground, the inlet can ingest ground debris and hot gasses from the fountain and ground vortex. This Hot Gas Ingestion (HGI) can cause a sudden loss of thrust (lift), impairing vehicle safety. The high-speed jet flow along the ground can also entrain the ambient flow, resulting in a low pressure region underneath the vehicle, leading to what is referred to as the “suckdown effect”.

\*Senior Research Scientist, Member AIAA

†Research Scientist, Associate Fellow AIAA

‡Research Scientist, Member AIAA

Copyright ©2002 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U. S. Code. The U. S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the copyright owner.

A number of numerical and experimental investigations have been carried out to better understand the complex time-dependent flows associated with powered-lift vehicles. One approach has been to simplify the geometry to study the basic flow physics. Van Dalsem et al. [2] performed time-dependent Reynolds-averaged Navier-Stokes (RANS) simulation using a delta wing with two aft mounted thrust-reverser jets in close proximity to the ground plane. These computations captured the loss of lift near the ground associated with the suckdown effect, and the small drop-off of lift at higher locations associated with the conventional ground cushion effect. Preliminary results were also presented for a Harrier YAV-8B forebody and inlet. Chawla and van Dalsem [3] carried out time-accurate laminar flow simulations using the same delta wing geometry of [2]. Static (fixed geometry) flow simulations were computed at varying heights above the ground plane, as well as a single maneuver simulation with the delta wing descending towards the ground plane. The static cases showed the expected trends between the lift coefficient and height, including the suckdown and cushion effects. The flows were found to be unsteady, with Strouhal numbers ranging from 0.015 to 0.03, and certain approximations had to be made in order to reduce the long compute times.

There have been several RANS computations using simplified Harrier geometries. Gea et al. [4] and Mysko et al. [5] both computed steady transonic flow (out of ground effect) about Harrier wing/fuselage configurations. Smith et al. [6] presented a time-dependent RANS solution about a simplified YAV-8B Harrier in ground effect. To date, this represents the only RANS solution about a fairly complete Harrier aircraft in ground effect. In order to offset the very long compute times, certain simplifications were made to the time-accurate approach. The current research expands on the work of Smith et al. by refining the geometry used in [6], calculating time-dependent solutions, and using these computed simulations to build a computational database.

The purpose of this paper is to describe a process that has enabled a static stability and control (S&C) database for the Harrier aircraft in ground effect to be computed. The focus of the current work is on the solution process itself, as opposed to the results of the simulations. A companion paper by Chaderjian et al. [7], contains results obtained using the process outlined in this work, including an analysis of 45 time-dependent RANS simulations at 5 different heights and 9 angles of attack. In order to compute a S&C database using time-dependent CFD simulations, it's necessary to automate as much of the process as possible, from mesh generation to post processing. This paper describes a strategy that allows this automation, as well as still providing the ability to easily extend and customize

the process without relying upon cumbersome software or third-party support. Another focus of the current work is the parallel efficiency of the flow simulation process, as an efficient flow solver is necessary in order to compute a time-dependent low frequency flowfield. The efficiency of the OVERFLOW-MLP solver, and it's interaction with the runtime environment are examined.

The first section describes the heterogeneous computing model that was followed, and the script system that was built to generate a time-dependent database. Next, the details of the computational mesh are presented, and the automatic generation of configurations for the different heights and angles of attack required to fill the S&C database is discussed. Lastly, the efficiency of the flow solver and the embarrassingly parallel solution strategy are described, followed by a description of the post-processing tools.

## 2 Remote Interface

Computing an S&C database involves performing a parametric study of the forces and moments on the aircraft. In this work, the parameters of interest were chosen to be the height of the vehicle above the ground plane, the angle of attack, and the freestream Mach number ( $h$ ,  $\alpha$ , and  $M_\infty$ ). Working with such a parameter space, it's desirable to automate as much of the solution process as possible - from mesh generation to post-processing of the computed results. This automation speeds the overall process and minimizes human errors. To this end, a modular script system was developed that allowed different facets of the solution process to utilize a consistent interface and set of tools. The script system was built from the "ground up", rather than from the "top down", i.e. by building the lowest level components first and then building application scripts on top of the low level components.

This script system was implemented using the Perl language. Perl was chosen because it is an object-oriented language which encourages the re-use of components, and it's a powerful interpreted language which can fulfill all of the needs of the script system without resorting to creating specialized compiled binaries, or mixing scripting languages. One of the requirements on the script system for the current work was to manage a heterogeneous, specialized computing environment. In this environment, each computer (or group of computers) has a specific dedicated task - one class of machine is dedicated to archival storage, one machine is dedicated to high-performance computing, one machine is dedicated to post-processing and flow visualization (cf. Fig. 2). This type of environment is at least conceptually similar to grid computing environments, such as NASA's Information Power Grid (IPG) (cf. [8]),

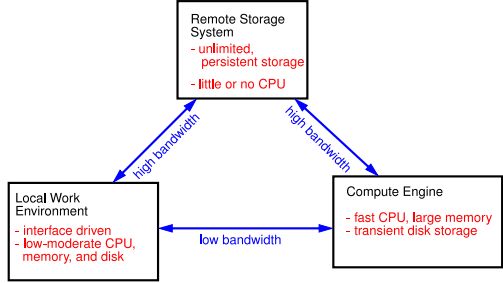


Figure 2: Schematic of computing environment.

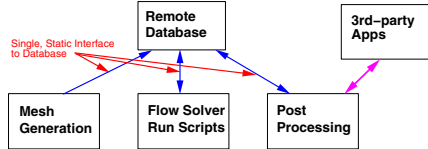


Figure 3: Schematic of script system.

as for any single simulation there is only one location where the data is stored and one location where the computation occurs (in most cases).

The backbone of the script system is an interface to an S&C database stored on a remote archival storage system. This type of storage paradigm assumes that the remote storage is the only system in the heterogeneous environment that can maintain persistent data. All interaction with the database utilizes this same interface (cf. Fig. 3). The interface to the remote database is implemented using a Perl module (object). In an object-oriented fashion, the interface is considered static and is the means by which clients (the application scripts in this instance) interact with the database. The actual implementation details behind the interface are hidden from clients and hence can be changed at any time, provided that the interface remains static. This allows the design to evolve with the needs and abilities of the rest of the system. For example, currently the database is implemented simply using a UNIX file and directory structure. If this was replaced with a more sophisticated database software tool, none of the application scripts would need to be changed. The application scripts are not even aware that the data is stored remotely. The interface currently supports several low-level methods, such as “copy-to”, “copy-from”, etc. from which higher-level tools can be developed. In practice, an application script would issue a command and provide the current parameter values ( $M_\infty$ ,  $\alpha$ ,  $h$ ), and the Perl implementation stores the file in the proper location within the database. All files are stored in the database as read-only to prevent multiple users or runaway processes from overwriting entries in the database.

Using the remote database module, scripts

were developed to generate the overset grid system, run the flow solver, and post-process the results through a GUI (written using Perl-Tk). In this manner, not only is there considerable software re-use among the various scripts, but an entirely new database can be accessed simply by creating a new specific instance of the remote database module, i.e. the higher-level scripts do not need to change to compute a new database. The complete script system contains about 1500 lines of Perl code, of which about 50% is dedicated to the GUI. The script system designed to generate the overset grid system is outlined in the next section, the scripts which run the flow solver are briefly described in Sec. 4, and the post-processing tools are discussed in Sec. 5.

### 3 Computational Mesh

Numerical simulation of the Harrier in ground effect combines the complex geometry of a full-aircraft configuration, and the complex physics of a jet in crossflow impinging on a ground plane. In order to accurately simulate this type of flowfield, Navier-Stokes simulations are required to resolve the viscous jet impingement on the ground plane, the interaction of the jet and ground vortex “fountain” with the aircraft, as well as the features of the jets in crossflow themselves (cf. Fig. 1). An overset grid strategy (cf. [9, 10]) was chosen due to the complex geometry and complex physics encountered. Using overset grids allows different regions of the flowfield, which require different levels of physical modeling, to be easily handled, and the relevant features of the geometry can be easily modeled. First an overview of the computational geometry and overset grid system is provided, then the details of the automation process required to generate separate grid systems for each aircraft configuration is presented.

#### 3.1 Overset Grid System

The initial definition of the Harrier YAV-8B geometry was obtained from the work of Smith et al. [6]. The definition used in [6] did not include several features of the aircraft, and these were added from the original “lofting-line” data supplied by Boeing (at the time the McDonnell Douglas Aircraft Company). Figure 4 shows the computational geometry used in the current study. Most of the major components of the aircraft are modeled, including the wing with leading-edge root extension (LERX), empennage, engine inlet region, and the two engine exhaust nozzles. The engine exhaust nozzles are scheduled 81 degrees from the aft position, and the engine thrust rating used for all calculations was “short-lift wet”, again following [6], where the nozzle exit conditions were determined from an engine-deck code.

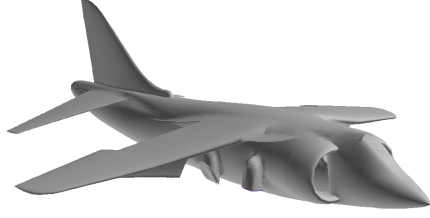


Figure 4: Computational geometry for Harrier YAV-8B

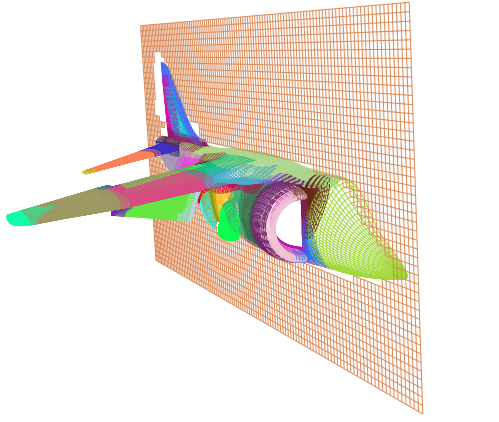


Figure 5: Harrier overset surface mesh definition

The positive circulation flaps are also modeled in their fully-deflected position. Due to time constraints, it was not possible to model the fuselage gun pod/lift-improvement devices (LIDS). Siuru [1] contains a discussion of the development and purpose of these lift-augmentation devices for the Harrier.

Viscous body-conforming grids were generated about the relevant features of the Harrier geometry, and the overlapping surfaces are shown in Fig. 5. The height of the first cell above the solid surface was specified such that  $y^+ \leq 5.0$  in each zone, which was found to be sufficient to resolve the viscous stresses on the surface during a takeoff or landing scenario. The Reynolds number is 12 million, based on the aircraft length. Currently sideslip is not considered, so all of the computational meshes take advantage of a lateral symmetry plane on the centerline of the body. An inviscid Cartesian grid, here called the “near-body Cartesian box”, was created to surround the viscous grids with little overlap, and portions of this Cartesian grid which were interior to the solid surface of the aircraft were removed using the overset hole-cutting procedure (cf. Fig. 5). This provided a region surrounding the aircraft which could be placed at any height and orientation relative to a ground plane, without affecting any of the intergrid connectivity between the viscous zones. In other words, the vis-

cous zones and near-body Cartesian box could be processed through the overset connectivity code (Pegasus 4.1 [11] was used for the current work) once, and then utilized for every configuration without any changes. All of the surface and volume meshes were created using the OVERGRID tool for overset grids developed by Chan [12]. The viscous body-conforming grids, along with the near-body Cartesian box comprise 45 zones and 2.4 million grid points.

### 3.2 Process Automation

Simulating the Harrier in ground effect requires proper treatment of the aircraft/ground plane interaction, which varies from the usual treatment of an aircraft in flight. As a first approximation, it’s been assumed that the parameters which affect a vehicle in ground effect are the height above the ground plane ( $h$ ), the angle of attack ( $\alpha$ ), and the freestream Mach number ( $M_\infty$ ). Fig. 6 shows the reference frame transformations that can be applied to an aircraft in ground effect. Figs. 6b and c would be suitable for a numerical simulation using a fixed grid system, as is used in the current work. Reference frame b) was chosen for this work, so that the flow visualization could take place in a natural frame of reference.

The viscous body-conforming region was combined with the automated script system developed by Rogers et al. for the High Wing Transport [13], in order to develop a system which could automatically generate the complete system for the aircraft and ground plane in combination. The script system allowed the processing to be performed in two steps; first the aircraft was placed at the desired height and orientation and the viscous intergrid connectivity was established, then the ground plane and farfield grids were placed around the aircraft, and the complete connectivity was determined. An example complete grid system is shown in Fig. 7. The jet exhaust region was modeled using viscous Cartesian zones which are generated specifically for each height and angle of attack. The viscous ground plane is broken into three regions to allow higher resolution directly under the aircraft where the jet impingement creates large gradients. The final grid system consists of 52 zones, and 3.6 million grid points for the configuration with the aircraft at 30 feet above the ground plane. The number of grid points changes with height, due to the generation of specific grids for the jet region at each height, as will be discussed below.

In order to avoid generating field grids specifically for each configuration, a general set of field grids were created and an overset hole-cutting procedure was used to eliminate portions of the grids which weren’t necessary, depending upon the aircraft height. For example, the inviscid



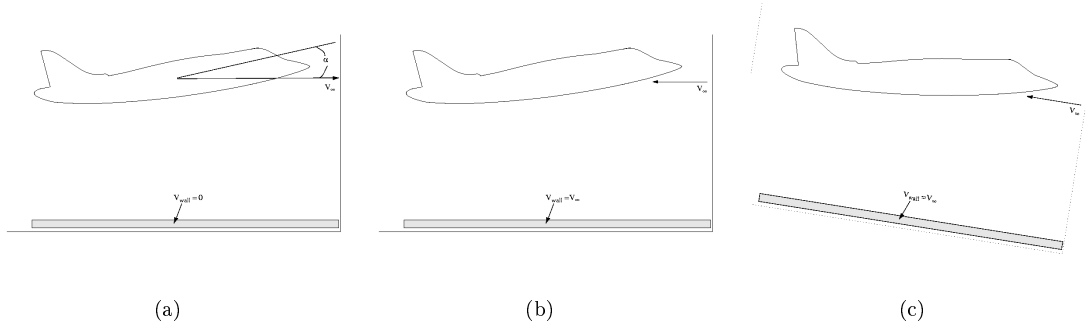


Figure 6: Aircraft frame of reference transformations including ground plane

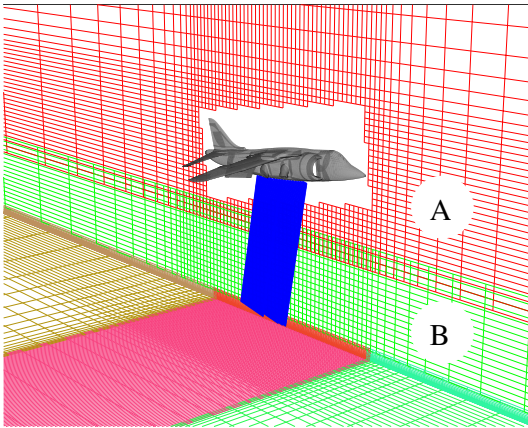


Figure 7: Viscous ground plane and inviscid outer field zones

grid labeled “B” in Fig. 7 actually extends above the aircraft, however that portion is removed by the the grid labeled “A” which surrounds the aircraft box. Similarly, depending upon the height and angle of attack, grid A and the jet grids may extend below the viscous ground plane, so a cutting plane is established to remove them if necessary. In this manner, some points aren’t utilized due to the hole cutting procedure, but there is no need to generate separate field grids for each configuration. The exception to this strategy is the viscous zones which model the jet convection to the ground plane. These zones are computed using a complete Navier-Stokes algorithm, as opposed to the thin-layer Navier-Stokes approximation which is used for the body-conforming viscous zones. Hence these jet grids are both dense and expensive to compute. Instead of removing portions of the jet grids using an overset hole-cutting procedure, which would lead to expensive computations essentially being thrown away due to the iblank logic, these grids were generated specifically for each height and then rotated

into position relative to the aircraft based on the angle of attack.

The wallclock time required to generate a single overset grid system, including storing the files on a remote database, was approximately 10 minutes. The processing system was a 2-CPU desktop workstation, and Pegasus 4.1 [11] was used as the overset grid processing software.

An alternative to pre-processing all of the mesh generation and overset grid connectivity information is to simply store some mesh generation “meta-information”, such as the input files, surfaces, etc., and then generate the meshes and connectivity “on-the-fly”. If we assume that a distributed computing environment (i.e. coast-coast file transfers) can sustain a throughput of 1 Mb/sec, then to transfer the 100 Mb mesh for the current problem would require approximately 100 sec. The volume mesh generation software generates about 100k cells/sec, and the mesh generation can be run in parallel since each overset grid is independent, so clearly generating the mesh is preferred to transfer. The overset grid connectivity data typically is about 25% the size of the mesh file, and hence would require about 25 sec to transfer. Domain connectivity can currently be performed in parallel, using Pegasus 5 [14] or Meakin’s DCF package [15], and would require about 30 sec. - 1 min. to process the current mesh. These numbers do scale approximately linearly with problem size. Note that this discussion does not even include any data migration time within a mass storage system, or parallel process inefficiencies incurred by the serial transfers. This indicates that for a distributed computing environment, generating the mesh on-the-fly as opposed to a priori mesh generation is an attractive paradigm.

### 3.3 Unsteady Flow Features

The current work does not attempt to provide a “best” single solution for the Harrier in ground effect, rather, the emphasis was to uti-

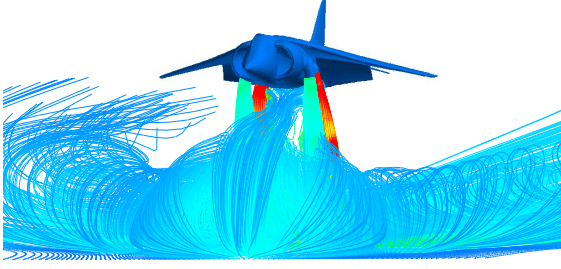


Figure 8: Snapshot of unsteady streaklines (colored by temperature) emanating from jet exhausts ( $h = 15$  ft.,  $\alpha = 6.0^\circ$ ,  $M_\infty = 0.05$ ).

lize the best grid system possible to calculate a time-dependent S&C database given the current computational resources. As such, the current grid system is very much an engineering compromise. A snapshot of the unsteady streaklines in the jet/ground plane flowfield is shown in Fig. 8 for a height of 15 ft. and  $\alpha = 6^\circ$  (compare with Fig. 1). The streaklines are colored by temperature, with red being hot and blue cold. The jet flow impinges upon the ground plane, and then spreads in all directions. The oncoming freestream flow causes the forward jet ground flow to separate and roll up into a large vortex under the aircraft. This vortex changes size and position with time, similar to the “puffing” behavior cited by Cimbalá et al. for an isolated jet in crossflow [16]. The time-dependent aerodynamic lift force on the aircraft is shown in Fig. 9. After an initial transient, an unsteady oscillation is evident, which corresponds to the change in ground vortex strength and position. The frequency of the lift oscillation is 2.22 Hz ( $St_{D_{jet}} = 0.04$ ), which is close to the unsteady oscillation measured for an isolated jet in [16], and also that computed for a delta wing in hover by Chawla and van Dalsem [3]. See the companion paper [7] for more details on the unsteady flowfield, and a discussion of the full unsteady database.

## 4 Simulation Process

### 4.1 OVERFLOW-MLP Solver

One of the goals of the current project is to utilize large-scale parallel computers, such as the NASA Ames 1024-processor R13000 SGI Origin 3000 (O3K) machine, to compute the S&C database. The SGI O3K is a shared-memory, multi-processor (SMP) machine. A version of the OVERFLOW solver [17], called OVERFLOW-MLP (for “multi-level parallelism”), has been developed by Taft [18] for use on these types of architectures. The OVERFLOW-MLP solver uti-

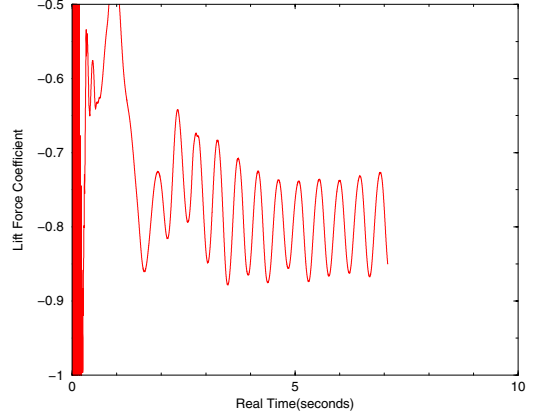


Figure 9: Time variation of lift history ( $h = 15$  ft.,  $\alpha = 6.0^\circ$ ,  $M_\infty = 0.05$ )

lizes two levels of parallelism - domain decomposition, and procedural (or loop-level) parallelism. The domain decomposition scheme is implemented on top of the “production” version of the parallel OVERFLOW solver.

As the OVERFLOW-MLP solver uses two methods of parallelization, there are combinations of parameters that can be varied to affect performance of the code. The main parameters are the number of processors per domain (which can vary from one domain to the next), the number of domains, and the total number of processors utilized. As a large number of cases are required to fill the S&C database, some experimentation was performed to determine an optimum configuration of the OVERFLOW-MLP control parameters for the current problem configuration.

Using the OVERFLOW-MLP code essentially as a pure domain-decomposition scheme (i.e. using 1 CPU per domain), the parallel efficiency was 99% when using 16 CPUs for the current Harrier application. This was suitable for computing a database using the embarrassingly-parallel approach to be discussed in the next section, however it is still desirable to utilize large numbers of CPUs in order to reduce debugging time, explore database-fill methods other than the brute force approach, and for future work simulating vehicle maneuvers. Figure 10 shows the parallel speedup when computing the Harrier configuration using a variety of domain sizes and number of CPUs. It’s seen that while some configurations can provide acceptable performance using up to 48 CPUs, none of the options performs well using more processors.

There are two contributing factors to the poor parallel efficiency with large numbers of processors; the scalability of the underlying procedural parallelism in the production OVERFLOW code, the low number of degrees of freedom for

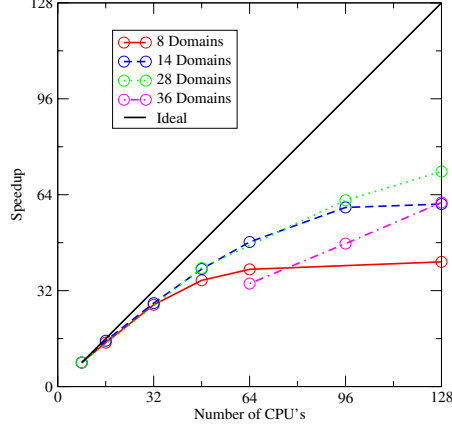


Figure 10: Parallel speedup for the OVERFLOW-MLP code computing the Harrier configuration (SGI R12000 O2K).

the load balance algorithm. For a multi-level parallelism scheme, such as the OVERFLOW-MLP code, the total parallel efficiency ( $\eta_T$ ) can be (approximately) viewed as a product of the domain-based efficiency ( $\eta_D$ ), and the procedural efficiency ( $\eta_P$ )

$$\eta_T \approx \eta_D \cdot \eta_P = \frac{wt_1}{wt_N \cdot N} \quad (1)$$

where  $wt_N$  is the walltime using  $N$  CPUs. Since  $\eta_T$  and  $\eta_D$  can be measured directly, the approximate procedural efficiency can be calculated. Reducing the data in Fig. 10 using Eqn. 1, leads to a procedural efficiency curve for the underlying algorithm for the Harrier application (cf. Fig. 11). Note that the average number of CPUs per domain is utilized as the abscissa. The procedural efficiency for the current problem is also measured directly by performing computations with a single domain. Note that this measurement only provides an approximate scaling, as the current problem requires between 4 and 8 CPU's of memory depending upon the machine being utilized. In Fig. 11, all of the computations are collapsing to the same curve, to within the variability of the timing data. Further, the procedural efficiency reduces quickly, so that with 4 CPUs/domain the procedural efficiency has already dropped to about 80% for this application. Note that this curve is only relevant to the current application. It's believed that for applications which primarily use the more "cache-friendly" thin-layer algorithm the scalability would improve.

As the number of domains is increased, the number of CPU's per domain decreases, while the load balance remains over 80%. This accounts for the improvement seen in Fig. 10 increasing from 8 to 28 domains. However as the number of domains increases it becomes more difficult to

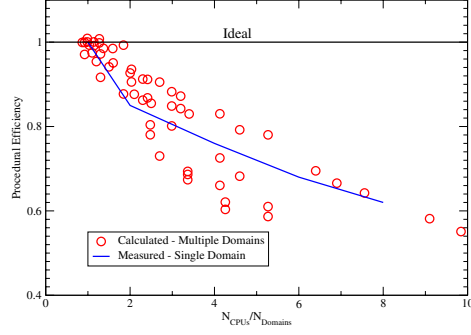


Figure 11: Procedural efficiency for the OVERFLOW-MLP code computing the Harrier configuration (SGI R12000 O2K and R13000 O3K).

achieve an efficient load balance, due to the limited number of degrees of freedom. The overall performance of the code again degrades, as seen with the speedup for 36 domains in Fig. 10. It is possible to achieve a good parallel efficiency for the current problem by trial-and-error. Following the above analysis, Pandya et al [19] manually split many of the larger grids, and by adjusting the load-balance weights were able to achieve acceptable efficiency using 112 processors and 32 domains with 67 overset grids. This is obviously not a general solution however.

Unstructured viscous solvers can achieve linear scalability up to 128 processors on problems of this size (cf. Mavriplis [20] and Anderson et al. [21]), in part because the number of degrees of freedom for the load balance algorithm is the total number of cells in an unstructured code, and in part because the underlying algorithms are performing more computational work at each iteration than the scalar, approximately-factored scheme utilized in the current work. In order to see similar performance with the OVERFLOW-MLP code for the current application, a means of providing more degrees of freedom to the load balance scheme is needed, perhaps by an automated adaptive blocking scheme for the inviscid outer flow such as used by Meakin [22]. Another approach would be to improve the scalability of the underlying procedural OVERFLOW code, or possibly extending the numerical scheme to take advantage of the large amount of memory that is currently unused when computing with large numbers of CPUs on large SMP machines (without sacrificing performance).

## 4.2 Script System

In order to compute an S&C database, it's desirable to take advantage of the "embarrassingly parallel" nature of the problem. In other words, since each configuration is an identical, independent problem with varying inputs, it's possible to compute several database parameters together in parallel and hence reduce the total wallclock time required to compute the entire database. For the Harrier application, the parameters of interest are height, angle of attack, and Mach number (cf. Sec. 3).

Two scripts were written to control the flow solver; the first runs the flow solver for a single set of parameters, while the second runs multiple concurrent sessions of the flow solver, each with a different set of parameters. Concurrent sessions (as opposed to multiple independent cases) were utilized in order to reduce the amount of job monitoring that needed to be performed. The main parameter affecting run time was found to be the height about the ground plane; lower heights required less iterations, while more were required for higher heights. An angle-of-attack sweep at a given height could be run concurrently, so that instead of monitoring hundreds of jobs individually, only a handful of AOA sweeps needed to be monitored\*. The script for running a single job (a job is a particular set of parameters, with a specific input file, flow solver, etc.) manages retrieving the latest flow solution from the remote database, setting up the input files, running the flow solver, checking if the flow solver executed properly, and then storing the new flow solution to the remote database. Since the flowfield in the current work is unsteady, this process must be iterated many times in succession. The locations of the various files (both remotely and on the local high-performance computing machine) and the flow solver inputs, are all determined based upon the parameters within the database matrix.

The script which runs multiple concurrent jobs leverages the work of the script which runs a single job. The multiple job script sets up a matrix of jobs to run, and then "launches" (literally `forks()`) an instance of the single job script for each set of parameters (cf. Fig. 12). The multiple job script then waits for each of these instances to finish, and when one does another job can be launched. When all of the jobs have finished the script exits.

When working on an SMP architecture it can be difficult to run multiple instances of the same executable concurrently on a dedicated set of processors without the multiple instances interfering with each other and greatly reducing the

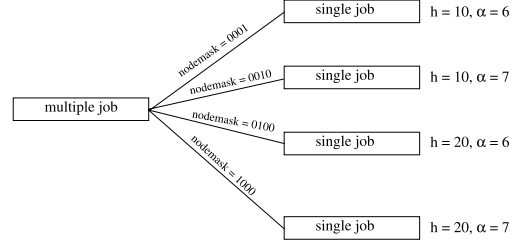


Figure 12: Schematic of running multiple concurrent jobs

efficiency of the parallel flow solver. The initial database computations (performed in March-April, 1999) had no way to account for processor interference, and hence the "embarrassingly-parallel efficiency" was not 100%. Rather, the average wallclock time required for a single database point when multiple concurrent computations were performed was roughly 1.3 times that required for a single computation in isolation.

On the NASA Ames Origin machines running the portable batch system (PBS), a "nodemask" variable is supplied to each batch job. This node-mask variable specifies the subset of the machines processors which the current batch session can access. The recent versions of the OVERFLOW-MLP code use the nodemask variable to execute a pin-to-node strategy, whereby the domain decomposition assigns domains to specific processors for the duration of the flow solver execution, rather than allowing the OS to dynamically balance the CPU usage. When running multiple concurrent versions of the OVERFLOW-MLP executable, it's thus necessary to apply another mask to the nodemask variable that PBS provides - a so-called "job mask" which specifies the processors which each specific job should utilize. In other words, if 16 processors are supplied by PBS, and it's desired to run 4 concurrent jobs, the job mask would mask portions of the original 16 processors so that 4 distinct processors are available to each job. This was done by creating a Perl module which stores the original PBS nodemask variable, and then portions it out to each job that is launched so that no two jobs can access the same processor (cf. Fig. 12). In this manner, concurrent versions of the same executable can be run without processor interference, and without requiring modification of the flow solver code. Using the job nodemask and the pin-to-node strategy, the embarrassingly-parallel efficiency when running multiple concurrent cases was indistinguishable from the theoretical maximum of 100%, however at the cost of a tight coupling between the flow solver and the run-time environment.

\*This approach is only practical on large machines such as the SGI Origins used in the current work. For a distributed computing platform, independent jobs would be required, and a more sophisticated job monitoring mechanism would be needed in the simulation process.

## 5 Post Processing

As with the mesh generation and flow simulation processes, when analyzing hundreds (or more) of CFD simulations it's desirable to automate much of the post-processing analysis. This automation can also be used to “mine” the database for interesting or critical points, as well as obtain general structures. As was discussed in Sec. 2, a GUI tool was built upon the remote database interface using Perl-Tk and utilized to perform post-processing analysis of the S&C database.

The GUI tool, referred to as “DBview”, has two main modes of operation; interactive and batch processing. In interactive mode DBview is itself in some sense simply a layer between the remote database and analysis application software residing on the engineers desktop machine (cf. Fig. 3). For example, DBview does not provide any graphical plotting capability, rather it retrieves the relevant data from the remote database and calls a third-party graphical application (Grace) to display and analyze the results. Similarly for flowfield visualization (FieldView) and frequency analysis (Matlab). In batch processing mode, a number of database entries can be selected to have certain tasks performed (with the results to be stored in the remote database), and the DBview tool again simply manages the data movement and choosing the proper application to run. With this approach, the GUI tool is both open-ended, since new third-party applications can simply be added as necessary, and lightweight, as the GUI simply must manage user input and data movement without needing to perform complicated or specialized analysis tasks. In other words, by leveraging these third-party applications, the post-processing tool can then be as powerful as an individual tailored tool, without having to support those features directly.

The DBview “front-end” is shown in Fig. 13. Access tabs for choosing the desired database are in the upper left of the interface. In this example, the two possible databases are “Baseline” and “SNC01”. The parameters shown in the selection area below the tabs change to those appropriate for the chosen database. Once the parameters are selected, the user can choose one or more of the check-buttons on the top right side of Fig. 13 to interrogate the grid and solution, or to view the load histories. Selecting the “grid” or “solution” check-button and pressing the “Get and Plot File” button results in the retrieval of the appropriate files from the database followed by an interactive flow visualization session. To aid the analysis, pre-loaded flowfield visualizations are available, and can either be printed to hardcopy or stored in the remote database. A sample of these “thumbnail” composite flowfield visualizations is shown in Fig. 14.

The GUI tool can take arbitrary slices through

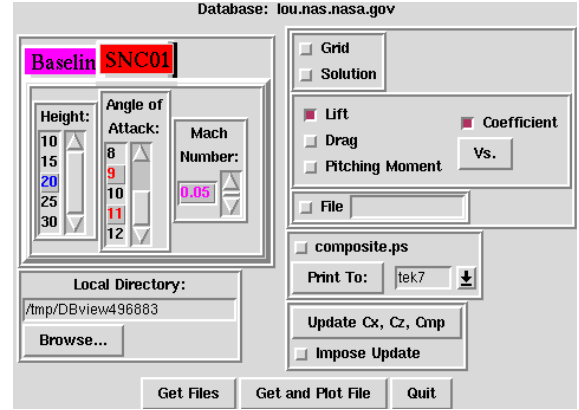


Figure 13: Database post-processing graphical user interface.

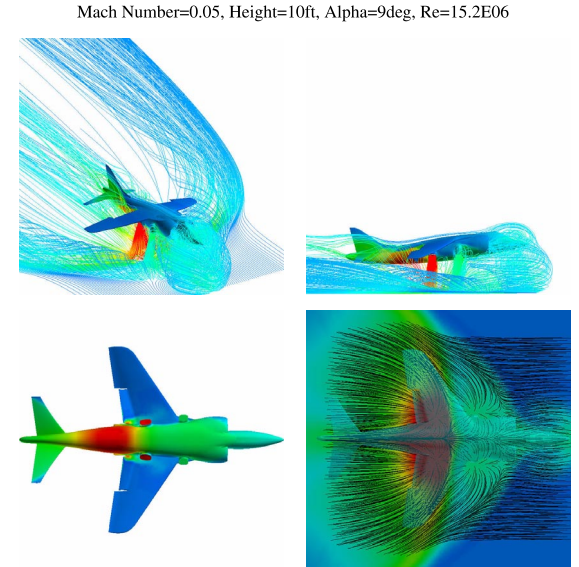


Figure 14: Sample of pre-loaded “thumbnail” composite flow visualizations. Color variations represent temperature.

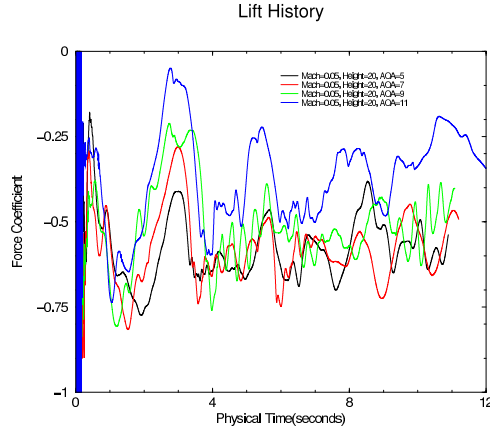


Figure 15: Sample of post-processing time-dependent load histories.

the data, including constant planes, vectors, or arbitrary isolated points in the database. The main interactive features of the tool are graphical plotting of the load histories\*, and 3-D visualization tools. An example of the load history post-processing is shown in Fig. 15. Arbitrary entries in the database can be selected and are automatically retrieved from the remote database and plotted in a third-party application. The labels and legends are also automatically generated. Additional curves can be added to the plot from the database as desired.

## 6 Summary

A method of automating the generation of a time-dependent, Navier-Stokes static S&C database for the Harrier aircraft in ground effect has been outlined. Lightweight reusable components were created which allow different facets of the CFD simulation process to utilize a consistent interface to a remote database. These components also allow changes and customizations to easily be facilitated into the solution process to enhance performance without relying upon third-party support. An analysis of the multi-level parallel solver OVERFLOW-MLP was presented, and the results indicate that it is efficient to utilize large numbers of processors ( $\approx 50$ ) even with a grid system with a relatively small number of cells ( $\approx 10^6$ ).

Two computational databases were generated using the methods described in this paper; one in Spring 1999 using a simplified Harrier geometry, and one in the Fall of 2000 using the geometry described here in Sec. 3. Table 1 contains a listing of the time required to obtain a single

\*All of the computed solutions are time-dependent, hence the time-history of the load variation must be plotted, not just a steady-state value.

time-dependent solution to 12 sec. of real time. This represents a worst-case scenario, and many of the computed cases require less time. It's seen that both the wallclock time to solution, and the amount of computational resources utilized were both reduced for the second database generation, i.e. the wallclock improvements are not just due to the use of more processors. The improvements in performance are due to many sources, however with the infrastructure now in place that allows the generation of time-dependent databases to be automated, improvements to the process can be easily implemented. In follow-on work that again utilizes the tools outlined in this paper, Pandya et al. [19] have extended this to improve the performance by almost an order of magnitude further in wallclock time.

Database	Wallclock time	Compute Time
Spring '99	30 days	5700 CPU-hours
Fall '00	8 days	3000 CPU-hours

Table 1: Time-to-solution for a worst-case, time-dependent Harrier computation.

## References

- [1] Siuru, W. D., *British Aerospace and McDonnell Douglas Harrier AV-8A/B*. Aero Publishers, Inc., 1985.
- [2] Van Dalsem, W. R., Chawla, K., Smith, M. H., and Abeloff, P. A., "Numerical Simulation of Powered-Lift Flows," in *International Powered Lift Conference Proceedings*, pp. III.16.1-14, The Royal Aeronautical Society, 1990.
- [3] Chawla, K., and Van Dalsem, W. R., "Numerical Simulation of a Powered-Lift Landing," AGARD-CP 534, 1993.
- [4] Gea, L. M., Chyu, W. J., Stortz, M. W., Roberts, A. C., and Chow, C. Y., "Flight Test and Numerical Simulation of Transonic Flow Around YAV-8B Harrier II Wing," AIAA Paper 91-1628, 1991.
- [5] Mysko, S. J., Chyu, W. J., Stortz, M. W., and Chow, C. Y., "Navier-Stokes Simulation of External/Internal Transonic Flow on the Forebody/Inlet of the AV-8B Harrier II," AIAA Paper 93-3057, 1993.
- [6] Smith, M.H., Chawla, K., and Van Dalsem, W. R., "Numerical Simulation of a Complete STOVL Aircraft in Ground Effect," AIAA Paper 91-3293-CP, July 1991.

- [7] Chaderjian, N.M., Pandya, S.A., Murman, S.M., and Ahmad, J., "Parametric Time-Dependent Navier-Stokes Computations for a YAV-8B Harrier in Ground Effect," AIAA Paper 2002-0950, Jan. 2002.
- [8] Johnston, W.E., Gannon, D., and Nitzberg, B., "Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid," in *Proceedings of the 8th International Symposium on High Performance Distributed Computing*, Aug. 1999.
- [9] Benek, J.A., Buning, P.G., and Steger, J.L., "A 3-D Chimera Grid Embedding Technique," AIAA Paper 85-1523, July 1985.
- [10] Buning, P.G., Chiu, I.T., Obayashi, S., Rizk, Y.M. and Steger, J.L., "Numerical Simulation of the Integrated Space Shuttle Vehicle in Ascent," AIAA Paper 88-4359-CP, Aug. 1988.
- [11] Suhs, N.E. and Tramel, R.W., "Pegsus 4.0 User's Manual," AEDC Tech. Report TR-91-8, 1991.
- [12] Chan, W.M., Meakin, R.L., and Potsdam, M.P., "CHSSI Software for Geometrically Complex Unsteady Aerodynamic Applications," AIAA Paper 2001-0593, Jan. 2001.
- [13] Rogers, S. E., Roth, K., Nash, S. M., Baker, M. D., Slotnick, J. P., Whitlock, M., and Cao, H. V., "Advances in Overset CFD Processes Applied to Subsonic High-Lift Aircraft," AIAA Paper 2000-4216, Aug. 2000.
- [14] Suhs, N.E., Rogers, S.E., and Dietz, W.E., "PEGASUS 5: An Automated Pre-processor for Overset-Grid CFD," Invited paper to be presented at AIAA Fluid Dynamics Conference, June 2002.
- [15] Meakin, R.L. private communication, Dec. 2001.
- [16] Cimbala, J.M., Billet, M.L., Gaublot, D.P., and Oefelein, J.C., "Experiments on the Unsteadiness Associated with a Ground Vortex," *AIAA Journal of Aircraft*, vol. 28, no. 4, pp. 261-267, 1991.
- [17] Buning, P.G., Jespersen, D.C., Pulliam, T.H., Chan, W.M., Slotnick, J.P., Krist, S. E., and Renze, K.J., *OVERFLOW User's Manual*. NASA.
- [18] Taft, J.R., "Multi-Level Parallelism, A Simple Highly Scalable Approach to Parallelism for CFD," in *HPCCP/CAS Workshop 98 Proceedings* (C. Schulbach, ed.), 1998.
- [19] Pandya, S.A., Chaderjian, N.M., and Ahmad, J., "Parametric Study of a YAV-8B Harrier in Ground Effect Using Time-Dependent Navier-Stokes Computations," submitted for publication at 20th AIAA Applied Aerodynamic Conference, June 2002.
- [20] Mavriplis, D.J., "Parallel Performance Investigations of an Unstructured Mesh Navier-Stokes Solver," ICASE Report No. 2000-13, Mar. 2000.
- [21] Anderson, W.K., Gropp, W.D., Kaushik, D.K., Keyes, D.E., and Smith, B.F., "Achieving High Sustained Performance in an Unstructured Mesh CFD Application," in *Proceedings of the Supercomputing Conference 1999*, Nov. 1999.
- [22] Meakin, R.L., "On Adaptive Refinement and Overset Structured Grids," AIAA Paper 97-1858-CP, June 1997.